International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Defending Future Phishing Website Assaults Using Machine Learning

Priya Shelke[1*], Riddhi Mirajkar[2], Dr. Vaishali Joshi[3], Jayashree Jankar[4], Prajkta Dandavate[5], Manisha Dabade[6]

Submitted: 29/01/2024 **Revised**: 07/03/2024 **Accepted**: 15/03/2024

Abstract: Phishing has become a more serious issue as a result of a significant increase in internet users. The phishing attacks of today constitute a serious threat to both the online environment and people's daily lives. In these attacks, the attacker poses as a reputable company in order to steal confidential information or the victim's digital identity, such as account login credentials, etc. A phishing website, also known as a faked website, imitates an official website's name and layout in an effort to fool users and steal their personal information. This study will present machine learning and deep learning techniques, and then use all of these algorithms to our dataset to identify fraudulent websites. The approach that works with the highest level of precision and accuracy, is then selected for phishing website detection. We hope that by putting in this effort, future phishing attacks may be better protected against.

Keywords: *MLP, XGBoost, Random Forest, ANN, SVM*

1. Introduction

There are more and more phishing incidents, which cost billions of dollars [1]. In these assaults, consumers enter sensitive information onto a forged website that seems real (such as passwords, credit card information, etc.). The most typical phishing targets are webmail and software-as-a-service (SaaS) websites [2]. The phisher creates websites that closely resemble the trustworthy ones. The fake links to the site is then distributed to countless internet users through emails and various other methods of contact. Emails, calls or instant messaging are frequently the source of these cyber-attacks [3]. In addition to stealing the identities, phishing attacks can also be used to spread additional software, such as exploit system weaknesses, or generate revenue [4]. The APWG report from 2020's third quarter states that there has been an increase in phishing assaults since March and around 30,000 distinct phishing sites have been discovered [2]. Again, in the same quarter, BEC assaults averaged a demand for $48,000, down from the second quarter's $80,000 and the first quarter's $54,000.

Because of the techniques used by the offenders to get past the anti-phishing systems already in use, researchers have a difficult time identifying and preventing phishing activities. In order to target some skilled and experienced users, the phisher can also use brand-new phishing schemes. Program-based methods are favoured to stop these attacks. Whitelist/blacklist [5], NLP [6], optical resemblance [7], rules [8], ML approaches [9], [10], etc. are the most common ways for identifying phishing attempts. Blacklist/whitelist-based techniques are ineffective in identifying unlisted phishing sites (also known as 0-h attacks), and they also fall short when a blacklisted URL is encountered with slight modifications. ML approaches train a model using a number of heuristic characteristics, including the web address, web pages' information, traffic from the website, browser, WHOIS data, and Site Rank, to boost detection efficacy. These heuristic traits, nevertheless, might be present on benign websites as well as phishing websites, which could lead to a misclassification. Furthermore, some of the heuristic features require third parties and are difficult to get. The visual identity of the website consists of screenshots, fonts, photos, page design, etc. As a result, these techniques frequently return false negatives and are unable to detect new phishing websites. The URL-based method may misread some dangerous websites hosted on public or compromised servers since it ignores the HTML of the webpage. The features of handcrafted URLs, such as the quantity of dots, the presence of special "@," "#," or "-" symbols, the URL's length, the location of the Top-Level Domain, the hostname's comparison to IP addresses, the existence of numerous TLDs, etc., are extracted by existing techniques [11], [12], [13]. There are still obstacles to collect manually URL characteristics, nevertheless, due to the time and increased personnel expenses required by human labour. Researchers face a huge issue in identifying and preventing phishing crimes since scammers utilize these offences in a way that can circumvent current anti-phishing tools. Thus, network security manager strongly advises against using a single strategy and in favour of hybrid methods.

[1,2,4] *Vishwakarma Institute of Information Technology, Pune, India*
[3] *KJEI's Trinity Academy of Engineering, Pune, India*
[5]*Vishwakarma Institute of Technology, Pune, India*
[6] *Walchand College of Engineering, Sangli, India*
[1]*ORCID ID : 0000-0002-5462-6530*
[2]*ORCID ID : 0000-0001-7284-9482*
[3]*ORCID ID : 0000-0002-1528-7193*
[4]*ORCID ID : 0000-0002-6169-9316*
[5]*ORCID ID : 0009-0006-5988-1916*
[6]*ORCID ID : 0009-0000-9038-317X*
** Corresponding Author Email: priya.shelke@viit.ac.in*

Four machine learning algorithms and two deep learning algorithms make up the algorithm presented in this work. The main objective of this algorithm is to correctly detect phishing websites.

The remaining text is arranged according to the format shown below. The paper's second section reviews the literature on phishing. The elements of the suggested methodology are explained in section 3. Section 4 presents the outcomes analysis and performance assessment. The confusion matrices of the models are compared and explained in Section 5. The conclusion is presented in Section 6.

2. Literature Review

Phishing websites, which prey on people's vulnerabilities rather than those of software, are a problem with internet security. It can be defined as a strategy used to lure internet users into divulging personal data, including passwords and usernames. In this work, Amani et al. [14] offer a clever technique for identifying phishing websites. The technique works as an extension that is added to a web browser to notify the user immediately when it comes across a phishing website. The technique is built using supervised learning, a type of machine learning. The Random Forest method was chosen because of its success in classification. By examining the traits of phishing websites and choosing the optimal collection of qualities to train the classifier with, they hope to create a classifier that performs better. They conclude their paper with an accuracy of 98.8% and a combination of 26 features as a consequence.

In this study, Jitendra et al. [15] looked at how well phishing URLs could be identified among a sample of URLs that included contained benign URLs. Additionally, they covered feature engineering, feature extraction utilising lexical analysis host-based features, statistical analysis, and randomization of the dataset. For the comparative investigation, they also tested a variety of classifiers, and discovered that the results are essentially consistent. They also noticed that dataset randomization produced a remarkable optimisation and that the classifier's accuracy greatly increased. They used straightforward regular expressions to take a simple approach to extracting the features from the URLs. More features might be tested, which could result in a further improvement in the system's accuracy. The URLs list in the dataset used in this study may be a little dated, so continued training on a new dataset on a regular basis would greatly improve the model's efficacy and precision. The main issue with the content-based strategy for identifying phishing URLs is the lack of phishing websites and the short lifespan of the phishing website, making it challenging to train an ML classifier based on its content-based features. As a result, they did not use content-based features in their experiment.

Internet users are subjected to a barrage of more sophisticated and frequent phishing assaults. Users who mistakenly provide their credentials to emails with websites that appear legitimate are jeopardising both their security and privacy. Methods like blacklisting these phishing websites grow ineffective and are unable to keep up with the explosion of fraudulent websites. Automated website detection must be able to keep up with this constantly changing type of social engineering. Hossein et al. [16] and produces current data on ML for phishing sites developed the “Fresh-Phish” technology. They use 30 distinct website features to create a sizeable-labelled dataset, which they then query using Python. They then test various ML classifiers on this dataset to see which is most precise. They looked at the effectiveness of the approach and the length of time model needed to be trained.

Numerous research has been conducted to investigate potential strategies to stop phishing attacks. In this study, Noor et al. [17] employ three machine learning techniques to detect phishing. Using a software solution that leverages URL analysis to discriminate between trustworthy and phishing websites, these models are trained using URL-based characteristics in an attempt to counter Zero-Day assaults. The random forest classifier performed from the data with 97% accuracy, 99% recall, and 97% F1 Score, respectively. The suggested method is rapid and efficient since, in contrast to past research, it only uses the URL for analysis and doesn't need any extra sources.

Three methods for spotting phishing websites are presented in the [18] paper. The first way involves looking at different URL attributes; the second entails confirming the legality of the website by learning where it is hosted and who is in charge of it; and the third method depends on visual inspection to ascertain whether a website is authentic. For the evaluation of these various URL and website properties, they apply machine learning techniques and algorithms. An overview of various strategies is also provided.

Keeping a black list of phishing URLs is the main defense against the behavior. A black list strategy, however, is reactive and unable to prevent newly discovered phishing websites. As a result, numerous studies have been conducted on the use of ML algorithms to identify previously unknown URLs used for phishing. Despite the fact that they yield good results, no such implementation has yet taken place. This is because of two things: 1) Not enough work has been done to provide a comprehensive end-to-end infrastructure for phishing URL detection; and 2) machine learning algorithms are too slow to identify phishing URLs. Farhan et al. [19] solved these two

problems by developing a strong framework for quick and automatic phishing URL identification. With a real dataset and a real-time configuration, they validated their framework and achieved an accuracy of 87%.

The prevalence of phishing attacks makes it important to employ adequate defence mechanisms and mount a successful defence against them. In order to minimise harm and increase public awareness of phishing attacks, Hesham et al. [20] seeks to identify phishing sites by an analysis of a three-component set. For each form of phishing attack, a countermeasure was suggested using website features based on the content analysis. To assess the efficiency of the countermeasure, these qualities will be categorised.as anti-phishing technology, the suggested solution improved site security. This work aimed to achieve maximum accuracy in phishing site recognition by combining three machine learning techniques into a single system: random forest, SVM, and decision tree. The proposed strategy outperformed the competition by 98.52%, according to the results.

In [21], the study reviews several phishing detection methods by dividing them into three major categories. The suggested model is then offered in two parts. In the first step, a number of machine learning algorithms are applied to assess the chosen dataset and apply feature selection methods to it. Consequently, using Random Forest for pairing 20 of the 48 features produced the best accuracy of 98.11%. In the second stage, many fuzzy logic techniques are implemented on the same dataset. Furthermore, the results of the studies carried out using fuzzy logic algorithms were excellent. When the FURIA algorithm was applied with just five features, the accuracy rate was 99.98%. Following that, a comparison and discussion are held on the outcomes of applying machine learning and fuzzy logic methodologies. When employing fuzzy logic methods, performance is better than when using machine learning algorithms.

The ensemble ML methods, Random Forest and XGBOOST are recommended by Dharani et al. [22] in this research to identify websites that are phishing depending on their Uniform Resource Locator. Phishing, which involves an attacker impersonating an already-existing, frequently- trusted individual or organisation in order to gain the target's account information, login details, and other private data, is one of the most widespread types of cybercrime in use today. Phishing sites resemble legitimate ones both aesthetically and linguistically. The frequency of phishing schemes has increased as a result of the increase in online trading activity. Since cybersecurity professions are the most difficult to fill, it is necessary to develop an automated method for phishing site identification. One of the most practical ways to solve this scenario is with machine learning because it can accurately classify the data and handle the changing nature of phishing schemes.

Phishing attempts, which steal user information from trustworthy websites, have been on the rise. Websites of persons or organisations are not the only targets of this form of attack. The numerous methods for phishing website detection that have been created using ML, DL, and various other methods still need to have their detection accuracy increased. Mohammed et al. [23] in this work provide an enhanced stacking ensemble approach for phishing website identification. The first ML algorithms whose parameters were optimised using a genetic algorithm (GA) were LightGBM, Bagging, AdaBoost, XGBoost, random forests, and GradientBoost. After grading the optimised classifiers, the best three models were chosen as the foundation classifiers of an ensemble stacking technique. The tests made use of three phishing website datasets: the UCI Data Set (Dataset 1), the Mendeley Dataset (Dataset 2), and another Mendeley Phish Sites Detection Datasets (Dataset 3). The results of the study showed that the improved ensemble stacking strategy improved detection accuracy, with values of 97.16%, 98.58%, and 97.39% for Datasets 1, 2, and 3, respectively.

3. Methodology

The module description of the analysis is represented by the framework in Figure 1.

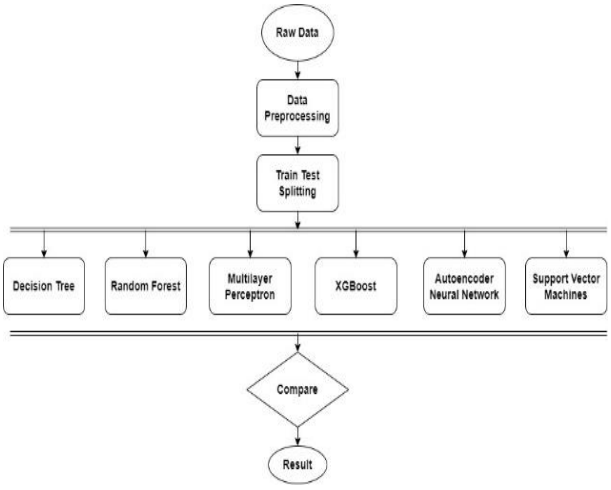


Fig. 1. Block Diagram

3.1. Data Collection:

The list of phishing URLs was generated using the open-source PhishTank program. This website offers a variety of phishing URLs that are updated hourly and come in several formats, including csv and json. This dataset contains 5000 randomly selected phishing URLs that are used to train machine learning algorithms. The official URLs can be found in the University of New Brunswick's open datasets. A selection of URLs that are not harmful, spammy, phishing, or defacing can be found in this dataset. Only the benign URL dataset is used in this study out of all of these sorts. This dataset contains 5000

randomly chosen genuine URLs that are used to train the machine learning models.

3.2. Feature Extraction:

Features are taken out of the URLs dataset in this step. The features that were extracted are divided into three categories:

- JavaScript and HTML based Features
- Address Bar Features
- Domain Features

3.2.1 Feature Based on the Address Bar:

Address bar fundamental characteristics may be retrieved from a wide variety of features. From among them, the following were picked for this project.

- Prefix or Suffix "-" present in Domain
- Using URL Shortening Services
- "http/https" present in Domain name
- Redirection "//" in URL
- URL Depth
- URL Length
- "@" Symbol present in URL
- IP Address present in URL
- URL Domain

3.2.2. Feature Based on Domain:

This category has several characteristics that may be extracted. Among them, the following were picked for this project.

- Traffic in Website
- DNS Record
- Domain - Age
- Domain – End Period

3.2.3. Feature Based on JavaScript and HTML:

This category has several characteristics that may be extracted. Among them, the following were picked for this project.

- Website-Forwarding
- Customization – Status Bar
- Redirection of IFrame
- Disable Right Click of mouse

3.3. Models and Training:

Before the ML model is trained, the data is divided into eight thousand training samples and two thousand testing samples. This clearly offers a challenge for supervised machine learning, according to the dataset. Supervised machine learning comprises two primary subcategories: classification and regression. The input URL can either be categorized as legitimate (0) or phishing (1), which creates a categorization problem for this data collection. The following supervised machine learning models (classification) are used to train the dataset in this research:

- MLP
- XGBoost
- Random Forest
- ANN
- Decision Tree
- SVM

The training dataset was used to construct each model, and the test dataset was used to evaluate them.

3.3.1. Decision Tree Classifier

Models for classification tasks frequently use decision trees. In essence, they use a sequence of if-then questions to arrive at a decision. Finding the set of if/else questions that leads us to the correct answer the quickest is the key to using a decision tree efficiently.

These types of queries are known as tests in the context of machine learning (as opposed to the test set, which is the collection of data we use to assess the generalizability of our model).

Before building a tree, the computer considers all possible tests and chooses the test that offers more insightful data about the variable being examined.

Measures for Attribute Selection: Choosing the ideal attribute for the root node and child nodes presents the largest issue when creating a decision tree. An strategy known as attribute selection measure can be used to overcome these problems. The optimal property for the tree nodes can be easily chosen using this measurement. The following are two well-liked ASM techniques:

Information Gain:

After dividing a data set based on a feature, information gain measures changes in entropy. This refers to how much knowledge a feature imparts about a class. We divide the node and construct the decision tree based on the significance of the information learned. In a decision tree, which constantly aims to maximize the value of information gain, the node with the biggest information gain gets split first. Equation (1) is applied:

$$\text{Information gain} = \text{Entropy(s)} - [(\text{Weighted Avg}) * \text{Entropy(each feature)}] \quad (1)$$

Entropy:

It is a statistic used to assess an attribute's impurity. Data randomness is defined by it. This is how entropy is calculated:

$$\text{Entropy(s)} = -\text{P(yes)log2 P(yes)} - \text{P(no)log2 P(no)} \quad (2)$$

where, as expressed in equation (2), "s" stands for the total number of samples, P(no) for the probability of no, and P(yes) for the probability of yes.

Gini Index:

The Gini index is a measure of purity or impurity used by the CART (Classification and Regression Tree) technique when building decision trees. A property with a low index is preferred over high value. The CART approach only produces binary splits, and it does so by employing the index. To figure out the, use the formula below:

$$\text{Gini Index} = 1 - \sum_{i=1}^{c}(Pi)^2 \quad (3)$$

where c is the no. of classes and Pi is the observed class's relative frequency in eq. (3).

3.3.2. Random Forest Classifier

It is now one of the most well-liked ML classification techniques. In essence, a random forest is an assemblage of distinct decision trees. Each tree in this scenario may provide somewhat accurate predictions, but it is likely to overfit certain data points.

By averaging the outcomes of several successful trees that show various types of overfitting, we can lessen the degree of overfitting. You can choose how many trees to include in your random forest model using the n_estimators option of the Random Forest Regression or Random Forest Classifier. They are usually very efficient, don't require data scaling, and function well with little parameter modification.

$$Gini = 1 - \sum_{i=1}^{c}(Pi)^2 \quad (4)$$

where c is the no. of classes and is the observed class's relative frequency.

The equation (4) for the Gini Index is calculated by deducting the sum of the squared probabilities for every class from 1. As a result, the class with the lowest Gini Index value receives the most priority because its likelihood of making a mistake is lower. Because it prefers big partitions and has an easy implementation, the Gini Index is useful.

3.3.3. Multilayer Perceptron's (MLPs): Deep Learning

MLPs, or multilayer perceptrons, are sometimes referred to as neural networks or (vanilla) feed-forward neural networks. It is possible to solve regression and classification problems with multilayer perceptrons.

MLPs are similar to generalized linear models in that they process information through multiple stages before coming to a decision.

Backpropagation: The Multilayer Perceptron's learning technique enables it to repeatedly alter the network's weights in an effort to reduce the cost function.

After the weighted sums are processed through all layers, the gradient of the Mean Squared Error is calculated for every pair of input and output in each iteration. The adjustment is then undone by adjusting the weights of the first hidden layer using the gradient value that was obtained. In this way, the weights are transmitted all the way back to the neural network's base!

$$\Delta_w(t) = -\varepsilon\frac{dE}{\mathrm{dw}(t)} + \alpha\Delta_{w(t-1)} \quad (5)$$

Where, $\Delta_w(t)$ is Gradient of Current iteration, -ε is the Bias, dE denotes Error. $\mathrm{dw}(t)$ expresses Weight vector, α shows learning rate, $\Delta_{w(t-1)}$ is Gradient of previous iteration shown in eq. (5).

3.3.4. XGBoost Classifier

It is one of the ML algorithms that is currently in use. The acronym for Extreme Gradient Boosting is XGBoost. A gradient-boosted decision tree implementation created for speed and efficiency is called XGBoost.

Mathematics behind XGBoost

Before discussing the math behind gradient boosting, here is a simple example of a CART that determines whether someone will enjoy fictitious computer game X. Here's a sample of a tree: The two decision trees aim to complement one another based on their total prediction scores, which is a crucial element of the example. Our model can be expressed numerically as follows:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F \quad (6)$$

where F is a set of potential CARTs, K is the number of trees, and f is the functional space of F in eq. (6). Therefore, objective function represented in eq. (7) as:

$$obj(\theta) = \sum_i^n l(y_i, \hat{y}_i) + \sum_{k=1}^{K}\Omega(f_k) \quad (7)$$

where, the second is the regularization parameter and first term is the loss function.

3.3.5. Autoencoder Neural Network

A neural network type with an equal number of input and output neurons is called an autoencoder. The input/output layers of the network include more neurons than the hidden layers. The autoencoder is trained to

encode the input data into the fewest possible hidden neurons. The predictors (x) and output (y) of an autoencoder are the same.

Here, brief explanation of the network's underlying mathematics because they are quite easy to comprehend. Encoders and decoders make up the two sections of the network eq. (10).

$$\phi : X \rightarrow F$$

$$\psi : F \rightarrow X$$

$$\phi, \psi = \underset{\phi,\psi}{\arg min} \|X - (\psi \circ \phi)\|^2 \quad (8)$$

The encoder function maps a bottleneck in the latent space (F) using the original data (X) as input. The decoder function (ψ) maps the bottleneck in the latent space (F) to the output. The input function and output function in this case are the same. In essence, we are trying to reconstruct the original image after applying some generalised non-linear compression.

The encoding network depicted in eq. (9), which passes through an activation function with z serving as the latent dimension, can be defined using the conventional neural network function.

$$z = \sigma(W_x + b) \quad (9)$$

Similar to the encoding network, but with differing weight, bias, and perhaps activation functions, the decoding network can be illustrated in eq. (10).

$$x' = \sigma'(W_z' + b) \quad (10)$$

The neural network will be trained using this loss function and the well-known backpropagation method. The network functions can be used to define the loss function (eq. (11).

$$\mathcal{L}(x, x') = \|x - x'\|^2 = \left\|x - \sigma'\left(W'\sigma(W_x + b)\right) + b'\right\|^2 \quad (11)$$

Given that the input and output images are the same, making it neither supervised nor unsupervised learning, this is frequently known to as self-supervised learning. The autoencoder's goal is to choose our encoder and decoder algorithms so that we just need the most basic data to encode a picture so that it may be created on the other side.

3.3.6. Support Vector Machines

Often called support-vector networks, support-vector machines (SVMs) are supervised learning models that assess data for classification and regression issues. Using a series of training samples, each of which is designated as falling into a different category, an SVM training technique creates a model that divides new occurrences into one of two categories. The method is a binary linear non-probabilistic classifier as a result.

Maximising the distance between the two hyperplanes is the main goal of SVM. The equation (12) is used to determine the margin:

$$m = \frac{2}{\|w\|} \quad (12)$$

The cost function now has a regularisation parameter as well. The regularisation parameter aims to strike a balance between margin maximisation and loss. After the regularisation parameter (eq. (13)) has been included, the cost functions appear as follows:

$$min_w \lambda \|w\|^2 + \sum_1^n \left(1 - y_i(x_i, w)\right)_+ \quad (13)$$

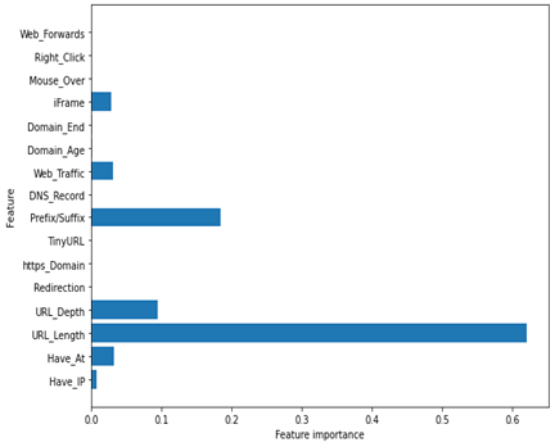
The loss function (eq. (14) known as hinge loss aids in maximising the margin:

$$c\big(x, y, f(x)\big) = \begin{cases} 0, & if\ y * f(x) \geq 1 \\ 1 - y * f(x), & else \end{cases} \quad (14)$$

4. Results

4.1. Decision Tree Classifier:

Feature importance in the model:

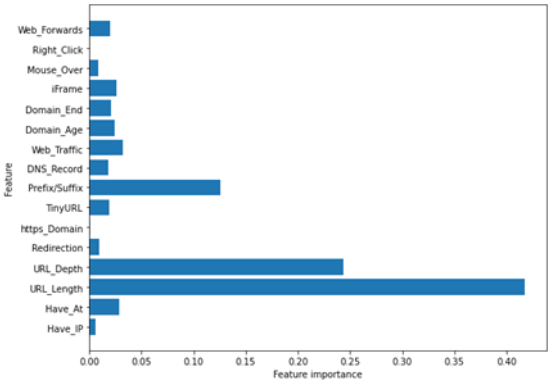


Decision Tree: Training Data - Accuracy: 0.810

Decision Tree: Testing Data - Accuracy: 0.826

4.2. Random Forest Classifier:

Feature importance in the model:



Random forest: Training Data - Accuracy: 0.814

Random forest: Testing Data - Accuracy: 0.834

4.3. Multilayer Perceptrons (MLPs): Deep Learning:

Multilayer Perceptrons: Training Data - Accuracy: 0.859

Multilayer Perceptrons: Testing Data - Accuracy: 0.863

4.4. XGBoost Classifier:

XGBoost: Training Data - Accuracy: 0.866

XGBoost: Testing Data - Accuracy: 0.864

4.5. Autoencoder Neural Network:

Autoencoder: Training Data - Accuracy: 0.819

Autoencoder: Testing Data - Accuracy: 0.818

4.6. Support Vector Machines:

SVM: Training Data - Accuracy: 0.798

SVM: Testing Data - Accuracy: 0.818

4.2 Comparison of Models:

To compare the models' performances, a data frame is constructed, with lists representing the outcomes of each model listed in Table 1.

Table 1. Comparison of different models

	ML Model	Train Accuracy	Test Accuracy
3	XGBoost	0.866	0.864
2	Multilayer Perceptrons	0.858	0.863
1	Random Forest	0.814	0.834
0	Decision Tree	0.810	0.826
4	AutoEncoder	0.819	0.818
5	SVM	0.798	0.818

The comparison above clearly shows that the XGBoost Classifier works effectively with this dataset.

5. Conclusion

Blacklisting and whitelisting have historically been used to mitigate these risks, nevertheless they can't detect phishing websites that aren't blocked (zero-day attacks), Thus, machine learning techniques are being applied as an enhancement to raise the accuracy of detection and lower the misclassification ratio. Attacks using phishing websites pose a severe risk to researchers and have been on the rise recently. However, some of them require information to be retrieved from sophisticated and challenging-to-use browsers, web traffic, and so forth. In this research, we train deep neural networks and ML models using the dataset developed for phishing website detection. By gathering both phishing and benign URLs of websites, a dataset is created that can be used to extract the essential URL- and website content-based properties. The performance level of each model is evaluated and compared. With a test accuracy of 0.864, we discovered that the XGBoost classifier, out of all others, performs best with the dataset being used.

Author Contributions

Jayashree Jankar: Conceptualization, Field study **Priya Shelke:** Methodology, Software, **Riddhi Mirajkar:** Data curation, Writing-Original draft preparation, **Vaishali Joshi, Jayashree Jankar:** Software, Validation., Field study **Prajkta Dandavate:** Visualization, Investigation, **Manisha Dabade, Priya Shelke**: Composing, editing, and reviewing.

Conflicts of Interest

Conflicts of interest are not disclosed by the authors.

References

[1] Aljofey, A., Jiang, Q., Rasool, A. et al. An effective detection approach for phishing websites using URL and HTML features. Sci Rep 12, 8842 (2022).

[2] APWG. Phishing Attack Trends Reports, 24, November 2020. https://docs.apwg.org/reports/apwg_trends_report_q3_2020.pdf (2020) (Accessed 14 January 2021).

[3] Aljofey, A., Jiang, Q., Qu, Q., Huang, M. & Niyigena, J.-P. An effective phishing detection model based on character level convolutional neural network from URL. Electronics 9, 1514 (2020).

[4] Dhamija, R., Tygar, J.D., & Hearst, M. Why phishing works. in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Montreal, QC, Canada, 22–27 April 2006, 581–590 (2006).

[5] Jain, A. K. & Gupta, B. B. A novel approach to protect against phishing attacks at client side using auto-updated white-list. EURASIP J. on Info. Security. 9, 1–11. https://doi.org/10.1186/s13635-016-0034-3 (2016).

[6] Sahingoz, O. K., Buber, E., Demir, O. & Diri, B. Machine learning based phishing detection from URLs. Expert Syst. Appl. 2019(117), 345–357 (2019).

[7] Haruta, S. , Asahina, H., & Sasase, I. Visual Similarity-based Phishing Detection Scheme using Image and CSS with Target Website Finder. 978-1-5090-5019-2/17/$31.00 ©2017 IEEE (2017).

[8] Cook, D. L., Gurbani, V. K., & Daniluk, M. Phishwish: A stateless phishing filter using minimal rules. in Financial Cryptography and Data Security, (ed. Gene Tsudik) 324, (Berlin, Heidelberg, Springer-Verlag, 2008).

[9] Jain, A. K. & Gupta, B. B. A machine learning based approach for phishing detection using hyperlinks

information. J. Ambient. Intell. Humaniz. Comput. https://doi.org/10.1007/s12652-018-0798-z (2018).
[10] Li, Y., Yang, Z., Chen, X., Yuan, H. & Liu, W. A stacking model using URL and HTML features for phishing webpage detection. Futur. Gener. Comput. Syst. 94, 27–39 (2019).
[11] Xiang, G., Hong, J., Rose, C. P. & Cranor, L. CANTINA+: a feature rich machine learning framework for detecting phishing web sites. ACM Trans. Inf. Syst. Secur. 14(2), 1–28. https://doi.org/10.1145/2019599.2019606 (2011).
[12] Zhang, W., Jiang, Q., Chen, L. & Li, C. Two-stage ELM for phishing Web pages detection using hybrid features. World Wide Web 20(4), 797–813 (2017).
[13] Rao, R. S., Vaishnavi, T. & Pais, A. R. CatchPhish: Detection of phishing websites by inspecting URLs. J. Ambient. Intell. Humanized Comput. 11, 813–825 (2019).
[14] Amani Alswailem; Bashayr Alabdullah; Norah Alrumayh; Aram Alsedrani, "Detecting Phishing Websites Using Machine Learning" (2019).
[15] Jitendra Kumar; A. Santhanavijayan; B. Janet; Balaji Rajendran; B.S. Bindhumadhava , "Phishing Website Classification and Detection Using Machine Learning" (2020).
[16] Hossein Shirazi; Kyle Haefner; Indrakshi Ray, "Fresh-Phish: A Framework for Auto-Detection of Phishing Websites" (2017).
[17] Noor Faisal Abedin; Rosemary Bawm; Tawsif Sarwar; Mohammed Saifuddin; Mohammd Azizur Rahman; Sohrab Hossain, "Phishing Attack Detection using Machine Learning Classification Techniques" (2020).
[18] Vaibhav Patil; Pritesh Thakkar; Chirag Shah; Tushar Bhat; S. P. Godse, "Detection and Prevention of Phishing Websites Using Machine Learning Approach" (2018).
[19] Farhan Sadique; Raghav Kaul; Shahriar Badsha; Shamik Sengupta, "An Automated Framework for Real-time Phishing URL Detection" (2020).
[20] Hesham Abusaimeh; Yusra Alshareef, "Detecting the Phishing Website with the Highest Accuracy" (2021).
[21] Almaha Abuzuraiq; Mouhammd Alkasassbeh; Mohammad Almseidin, "Intelligent Methods for Accurately Detecting Phishing Websites" (2020).
[22] Dharani M; Soumya Badkul; Kimaya Gharat; Amarsinh Vidhate and Dhanashri Bhosale, "Detection of Phishing Websites Using Ensemble Machine Learning Approach" (2021).
[23] Mohammed Al-Sarem; Faisal Saeed, "An Optimized Stacking Ensemble Model for Phishing Websites Detection" (2021).